BAB II
KAJIAN LITERATUR

2.1 Sistem Informasi

Sistem informasi dapat didefinisikan sebagai suatu rangkaian komponen yang saling terkait dan berinteraksi untuk mengumpulkan, menyimpan, mengolah, dan menyebarkan informasi guna mendukung pengambilan keputusan di dalam suatu organisasi [1]. Sistem informasi tidak hanya mencakup perangkat keras dan perangkat lunak, tetapi juga melibatkan pengolahan data, proses bisnis, dan partisipasi pengguna dalam penggunaan sistem. Hal tersebut menjadi landasan penting dalam pemahaman tentang fungsi dan peran sistem informasi dalam konteks organisasi *modern* [4]. Dengan demikian, sistem informasi merupakan alat yang vital bagi organisasi dalam mendukung kegiatan operasional dan pengambilan keputusan.

Sistem informasi memiliki beberapa komponen utama, berikut komponen-komponen utama dalam sistem informasi [1]:

1. *Hardware*

   Hardware merupakan komponen fisik yang menyusun sebuah sistem informasi. Bagian dari perangkat keras ini termasuk *workstation*, jaringan, *server*, kabel serat optik, perangkat seluler, pemindai, dan berbagai infrastruktur teknologi lainnya. Sebuah pusat data adalah contoh di mana banyak komputer terhubung dalam jaringan untuk memproses dan menyimpan data.

2. *Software*

   Software merupakan kumpulan program yang bertugas mengontrol perangkat keras komputer dan menghasilkan hasil atau informasi yang diinginkan oleh pengguna. Ini mencakup sistem operasi, aplikasi bisnis, dan berbagai program utilitas yang mendukung berbagai fungsi penggunaan komputer. Dalam konteks sistem informasi, perangkat lunak memiliki peran penting dalam mengelola data, menyediakan layanan kepada pengguna, dan menjalankan proses bisnis yang diperlukan.

3. Data

   Data merupakan materi mentah yang diolah oleh sistem informasi menjadi informasi yang berguna [4]. Sebagai contoh, dalam sistem informasi yang menggunakan basis data relasional, data disimpan dalam berbagai tabel. Dengan mengaitkan tabel-tabel tersebut, sistem dapat menampilkan informasi spesifik yang dibutuhkan pengguna, tanpa

kelebihan atau kekurangan. Pengguna dapat memilih untuk menampilkan satu atau semua item data serta menyaring data sesuai dengan batasan yang ditetapkan [4]. Sebagai contoh, pengguna dapat meminta daftar karyawan yang tinggal di kota tertentu dan bekerja lebih dari empat puluh jam dalam periode penggajian terakhir.

4. *Processes*

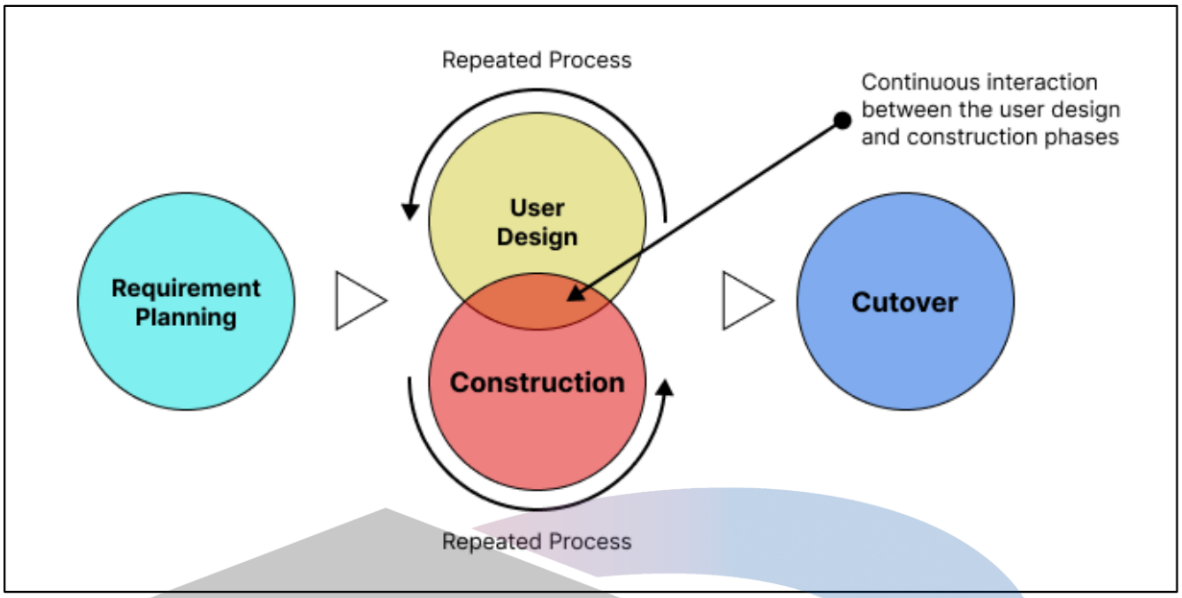
   Processes merupakan serangkaian tugas dan fungsi bisnis yang dilakukan oleh pengguna, manajer, dan karyawan teknologi informasi untuk mencapai hasil tertentu. Mereka menjadi bagian fundamental dari sistem informasi karena merepresentasikan operasi bisnis yang sebenarnya dalam konteks kegiatan sehari-hari. Untuk membangun sistem informasi yang efektif, pemahaman yang mendalam tentang proses bisnis diperlukan, serta dokumentasi yang cermat terhadapnya.

5. *People*

   People merupakan sekelompok individu yang memiliki kepentingan dalam sebuah sistem informasi atau sering disebut dengan *stakeholder*. *Stakeholder* yang dimaksud merupakan tim manajemen yang bertanggung jawab atas sistem, pengguna atau sering disebut sebagai *end-user* baik di dalam maupun di luar perusahaan yang akan berinteraksi dengan sistem, serta karyawan lain seperti analis sistem, programmer, dan administrator jaringan yang terlibat dalam pengembangan dan pemeliharaan sistem.

2.2 *Rapid Application Development* (RAD)

Rapid Application Development (RAD) merupakan sebuah pendekatan pengembangan perangkat lunak yang bertujuan untuk mempercepat proses pengembangan dengan fokus pada penggunaan siklus pengembangan yang lebih pendek dan iteratif [1]. Pendekatan ini berbeda dengan metode pengembangan perangkat lunak tradisional yang menerapkan siklus pengembangan yang lebih panjang dan terstruktur, seperti metode *Systems Development Life Cycle* (SDLC).



Gambar 2.1 Fase metode pengembangan *Rapid Application Development* (RAD) [1]

Berdasarkan gambar di atas, RAD memiliki 4 fase kerangka kerja utama. Adapun 4 fase kerangka kerja tersebut sebagai berikut [1]:

1. Perencanaan Kebutuhan (*Requirement Planning*)

   Tahap ini dimulai dengan identifikasi tujuan proyek dan pemahaman mendalam tentang kebutuhan pengguna. Pengembang dan perusahaan berkolaborasi untuk menetapkan cakupan proyek, menentukan fitur utama, dan merumuskan persyaratan fungsional dan non-fungsional.

2. Desain Sistem (*User Design*)

   Setelah kebutuhan di klarifikasi, pengembang memulai proses merancang antarmuka pengguna yang intuitif dan menarik. Mereka menggunakan teknik seperti *wireframing* dan *mockup* untuk membuat prototipe visual dari aplikasi. Prototipe ini kemudian diberikan kepada pengguna potensial untuk mendapatkan umpan balik terkait navigasi, tata letak, dan kegunaan umumnya. Berdasarkan umpan balik tersebut, desain prototipe diperbaiki dan ditingkatkan untuk mencapai pengalaman pengguna yang optimal.

3. Pengembangan (*Construction*)

   Setelah prototipe desain disetujui, tim pengembangan mulai membangun aplikasi secara aktif. Mereka mengimplementasikan fitur-fitur yang telah ditetapkan dalam tahap perencanaan, menggunakan teknologi dan alat yang sesuai. Proses pembangunan dilakukan dalam iterasi singkat atau *sprint*, di mana setiap iterasi menghasilkan potongan fungsionalitas yang dapat diuji secara terpisah. Setelah setiap iterasi selesai,

aplikasi diuji secara menyeluruh untuk mengidentifikasi bug dan masalah kinerja. Umpan balik yang diterima dari pengujian dan pengguna digunakan untuk memperbaiki dan memperbarui aplikasi dalam iterasi berikutnya.

4. Implementasi (*Cutover*)

   Setelah semua fitur telah dibangun dan diuji dengan baik, tahap pergantian dimulai. Pada tahap ini, semua komponen aplikasi diintegrasikan menjadi satu kesatuan yang utuh. Ini melibatkan pengujian integrasi, di mana aplikasi diuji untuk memastikan bahwa semua bagian berfungsi dengan baik bersama-sama. Selanjutnya, aplikasi disiapkan untuk peluncuran, termasuk pemasangan dan konfigurasi di lingkungan produksi. Sebelum diluncurkan, aplikasi juga harus melewati serangkaian pengujian akhir untuk memastikan kualitas dan keandalan nya. Setelah aplikasi diluncurkan, pengembang juga dapat melanjutkan pemeliharaan dan pembaruan berkelanjutan berdasarkan umpan balik dari pengguna.

Tujuan utama dari semua pendekatan RAD adalah untuk mengurangi waktu pengembangan dan perluasan dengan melibatkan pengguna pada setiap fase pengembangan sistem [1]. Dengan proses yang berkelanjutan, RAD memungkinkan tim pengembangan untuk melakukan modifikasi yang diperlukan dengan cepat seiring perkembangan desain [1]. Dalam konteks ketat nya anggaran perusahaan, penting untuk membatasi biaya perubahan yang biasanya terjadi dalam jadwal pengembangan yang panjang dan berlarut-larut.

Adapun beberapa keunggulan dalam menggunakan metode pengembangan RAD yaitu [5]:

1. Pengujian dan pelatihan menjadi lebih terintegrasi dalam proses pengembangan karena merupakan bagian alami dari pendekatan *prototyping*.
2. Mendorong partisipasi aktif pengguna dan manajemen, meningkatkan antusiasme pengguna pada tahap akhir proyek.
3. Pengguna dan manajemen dapat melihat solusi berbasis perangkat lunak yang dikembangkan lebih cepat daripada menggunakan pendekatan pengembangan *model-driven*.
4. Kecenderungan *error* dan kehilangan data lebih mudah dideteksi dalam prototipe daripada dalam model sistem.

5. Pendekatan berulang menjadi lebih alami karena perubahan diharapkan dan dapat diakomodasi dengan fleksibilitas selama pengembangan.
6. Proyek mendapatkan visibilitas dan dukungan yang lebih tinggi karena keterlibatan yang luas dari pengguna selama proses pengembangan.
7. Bermanfaat untuk proyek di mana kebutuhan pengguna tidak jelas dan tidak pasti.

Sedangkan, terdapat pula beberapa kelemahan dalam menggunakan metode pengembangan RAD yaitu [5]:

1. Prototipe berbasis RAD mungkin membuat para analis merasa tidak percaya diri untuk mempertimbangkan alternatif-alternatif teknis lain yang lebih bernilai.
2. Ada kalanya sebuah prototipe lebih baik dibuang, tetapi para stakeholder sering enggan melakukannya karena mereka menganggapnya sebagai hilangnya waktu dan usaha dalam produk saat ini.
3. Adanya pendapat bahwa RAD mendorong mentalitas mengembangkan, mengimplementasi, dan memperbaiki yang meningkatkan biaya seumur hidup yang diperlukan untuk mengoperasikan, mendukung, dan merawat sistem.
4. Penekanan pada kecepatan dapat berdampak buruk terhadap kualitas yang disebabkan jalan-jalan pintas yang disarankan dengan buruk dalam metodologi tersebut.
5. Prototipe-prototipe RAD mungkin dapat memecahkan masalah yang salah karena analisis masalah disingkat atau diabaikan.

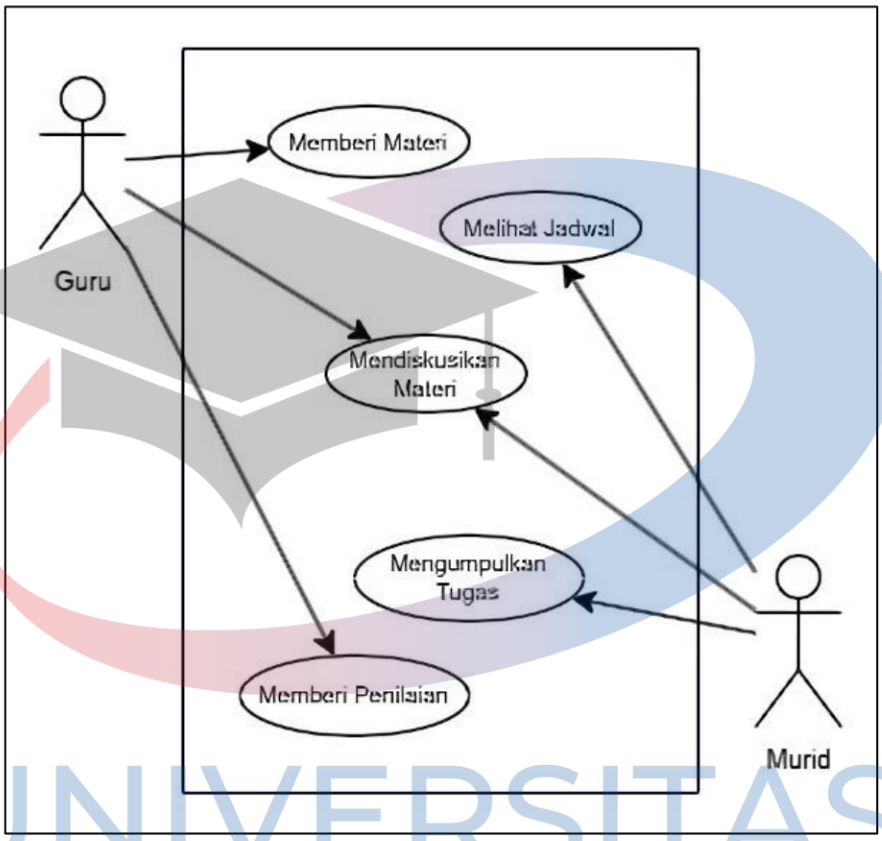
2.3 Teknik Pengembangan Sistem

2.3.1 *Use Case Diagram*

Use Case Diagram adalah alat visual yang digunakan untuk menggambarkan interaksi antara sistem dengan sistem eksternal serta pengguna. Dengan kata lain, diagram ini memberikan gambaran tentang siapa saja yang akan menggunakan sistem dan bagaimana cara pengguna mengharapkan untuk berinteraksi dengan sistem tersebut. Dengan menggunakan diagram ini, pengembang dan pemangku kepentingan dapat lebih mudah memahami tujuan dan skenario penggunaan sistem yang dirancang [1].

Misalnya, pada sebuah departemen layanan mobil terdapat berbagai aktor yang terlibat, seperti pelanggan, penulis layanan yang bertanggung jawab untuk menyiapkan pesanan kerja dan faktur, serta mekanik yang melakukan pekerjaan sesungguhnya pada kendaraan. Pada skenario tersebut *Use Case Diagram* akan menggambarkan bagaimana aktor-aktor yang berbeda berinteraksi dengan sistem.

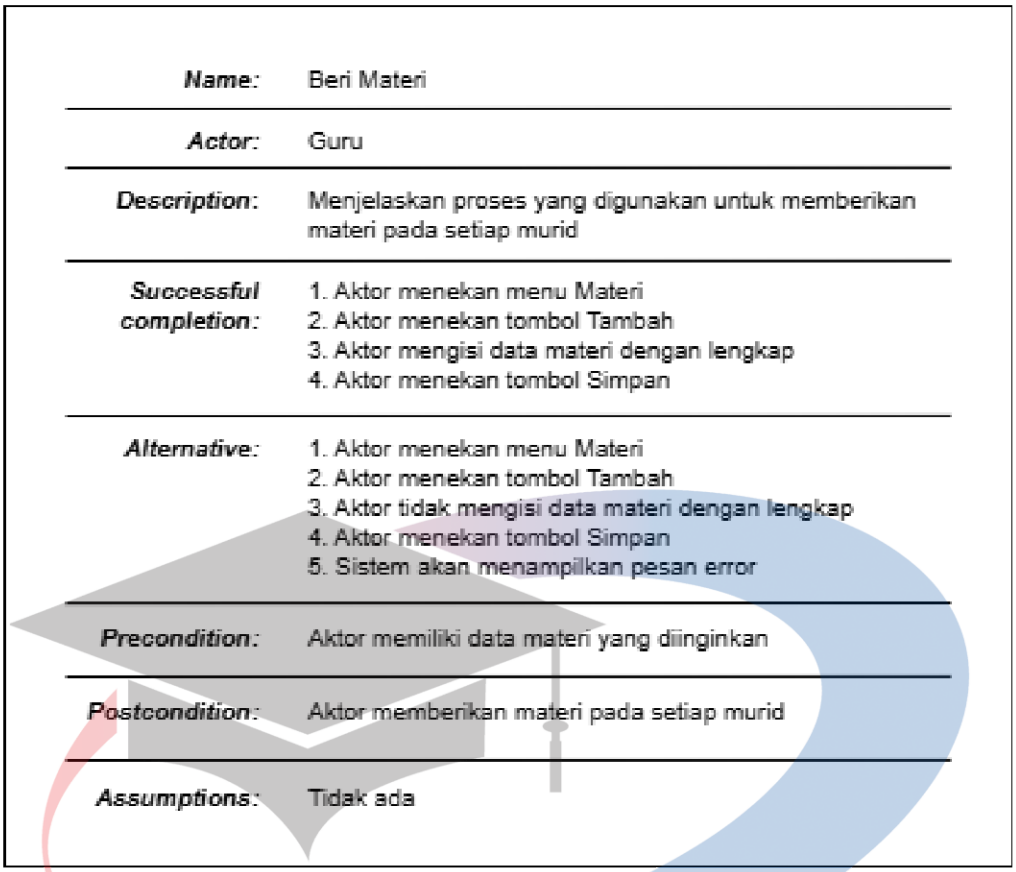
Saat membuat *Use Case Diagram*, langkah pertama adalah mengidentifikasi batas sistem, yang dimana digambarkan dengan sebuah persegi panjang [1]. Batas sistem menunjukkan apa yang termasuk dalam sistem digambarkan di dalam persegi panjang dan apa yang tidak termasuk dalam sistem digambarkan di luar persegi panjang, lalu langkah berikutnya adalah menempatkan *Use Case* pada diagram, menambahkan aktor, dan menunjukkan hubungan antara mereka [1].



Gambar 2.1 Contoh *Use Case Diagram* [1]

2.3.2 Deskripsi *Use Case*

Deskripsi *Use Case* adalah dokumen yang lebih merinci tentang interaksi antara pengguna atau aktor dengan sistem untuk mencapai tujuan tertentu. Hal ini mencakup berbagai elemen penting seperti nama *Use Case*, aktor yang terlibat, dan deskripsi singkat tentang fungsinya. Deskripsi ini juga merinci langkah-langkah atau tindakan yang perlu dilakukan untuk menyelesaikan *Use Case* dengan sukses. Langkah-langkah ini diatur secara logis dan mencakup baik tindakan aktor maupun respons sistem.



Name:	Beri Materi
Actor:	Guru
Description:	Menjelaskan proses yang digunakan untuk memberikan materi pada setiap murid
Successful completion:	1. Aktor menekan menu Materi 2. Aktor menekan tombol Tambah 3. Aktor mengisi data materi dengan lengkap 4. Aktor menekan tombol Simpan
Alternative:	1. Aktor menekan menu Materi 2. Aktor menekan tombol Tambah 3. Aktor tidak mengisi data materi dengan lengkap 4. Aktor menekan tombol Simpan 5. Sistem akan menampilkan pesan error
Precondition:	Aktor memiliki data materi yang diinginkan
Postcondition:	Aktor memberikan materi pada setiap murid
Assumptions:	Tidak ada

Gambar 2.2 Contoh Deskripsi *Use Case* [1]

Pada contoh gambar di atas, terdapat beberapa deskripsi atau keterangan seperti berikut:

1. *Name*

   Name menjelaskan tentang judul atau identifikasi dari *Use Case* yang akan dijelaskan.

2. *Actor*

   Actor mengidentifikasi siapa saja yang berinteraksi dengan sistem dalam konteks *Use Case* yang akan dijelaskan.

3. *Description*

   Description memberikan penjelasan singkat dan padat mengenai tujuan dan fungsi dari *Use Case*. Ini memberikan gambaran umum tentang apa yang dilakukan *Use Case*.

4. *Successful Completion*

   Successful Completion menjelaskan langkah-langkah dan tindakan yang diperlukan untuk menyelesaikan *Use Case* dengan sukses. Ini adalah urutan tindakan yang dilakukan oleh aktor dan respons dari sistem yang memastikan bahwa tujuan *Use Case* tercapai.

5. *Alternative*

   Alternative menggambarkan jalur alternatif yang mungkin terjadi jika kondisi normal tidak terpenuhi. Ini mencakup skenario-skenario yang berbeda dari alur utama dan bagaimana sistem menanganinya.

6. *Precondition*

   Precondition mencantumkan kondisi-kondisi yang harus terpenuhi sebelum *Use Case* dapat dimulai.

7. *Postcondition*

   Postcondition menjelaskan kondisi yang diharapkan setelah *Use Case* selesai.

8. *Assumptions*

   Assumptions mencantumkan asumsi-asumsi yang diambil selama penentuan dan pelaksanaan *Use Case*. Asumsi ini dapat mencakup keterbatasan, ketergantungan pada sistem lain, atau perilaku pengguna yang diharapkan.

2.3.3 *Class Diagram*

Class Diagram merupakan sebuah alat visual yang membantu memahami hubungan statis antara berbagai macam jenis objek atau elemen dalam sebuah sistem. Dengan menggunakan *Class Diagram*, pemahaman tentang struktur sistem secara keseluruhan dapat diperoleh, termasuk bagaimana setiap komponen saling terkait [1]. Konsep ini mirip dengan peta konseptual yang memungkinkan untuk melihat gambaran keseluruhan tanpa fokus terlalu mendetail pada suatu hal tertentu.

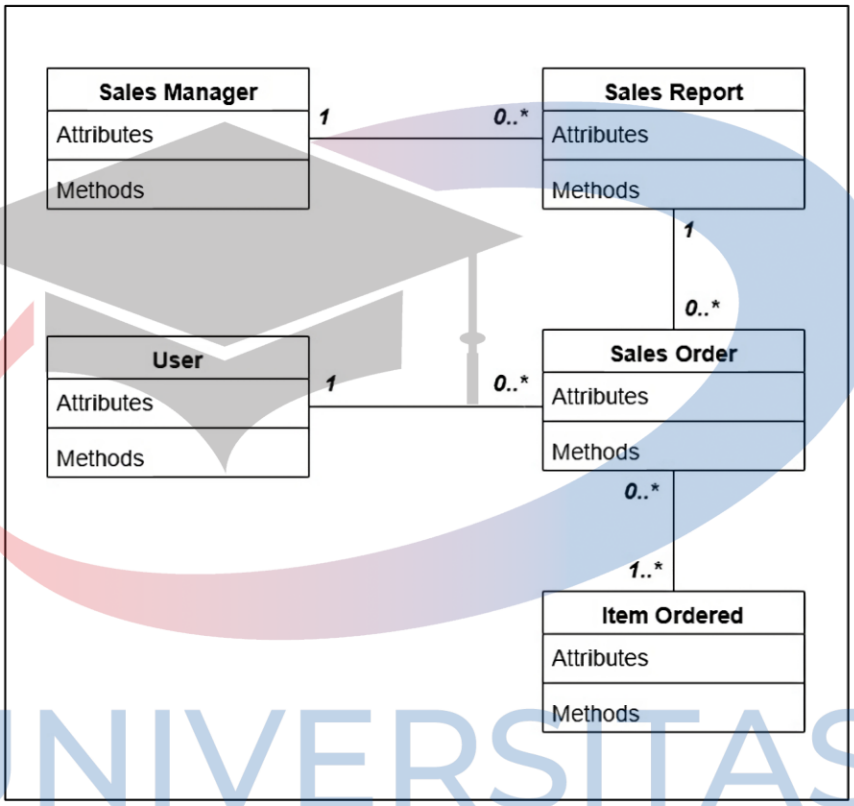
Melalui *Class Diagram*, informasi tentang kelas-kelas, atribut, dan metode mereka dapat disajikan dengan jelas. Hal ini dapat membantu peneliti atau pengembang perangkat lunak untuk memahami bagaimana objek-objek dalam sistem berinteraksi dan bertukar informasi satu sama lain [6].

Berikut beberapa fungsi utama dalam *Class Diagram*, yaitu [6]:

1. *Class Diagram* memperlihatkan struktur dari suatu sistem dengan jelas, memvisualisasikan hubungan antara kelas-kelas dan elemen-elemen lainnya.
2. Meningkatkan pemahaman tentang gambaran umum atau skema dari suatu program dengan memberikan representasi grafis tentang bagaimana kelas-kelas dan komponen sistem berinteraksi.
3. Dapat digunakan untuk analisis bisnis dengan menyajikan model sistem dari sudut pandang bisnis, membantu dalam memahami proses dan entitas yang terlibat dalam operasi bisnis

4. Memberikan gambaran menyeluruh mengenai sistem atau perangkat lunak serta hubungan-hubungan yang ada di dalamnya, membantu pengembang dalam merancang, memodifikasi, atau memahami arsitektur sistem dengan lebih baik.

Dalam *Class Diagram*, setiap kelas dapat di gambarkan dengan bentuk persegi panjang, dengan nama kelas di bagian atas, diikuti oleh atribut dan metode kelas. Garis-garis menunjukkan hubungan antara kelas-kelas dan memiliki label yang mengidentifikasi tindakan yang menghubungkan dua kelas tersebut [1].



Gambar 2.3 Contoh *Class Diagram* [1]

2.4 Basis Data

Basis data merupakan kumpulan data yang terorganisir secara sistematis dan disimpan secara terpusat. Data-data ini dapat diakses, dimanipulasi, dan dikelola sesuai kebutuhan aplikasi yang memanfaatkannya. Data dalam basis data direpresentasikan dalam struktur yang terstruktur seperti tabel dalam model relasional [7].

Basis data memungkinkan penyimpanan, pengelolaan, dan manipulasi data dengan efisien, serta menyediakan mekanisme untuk menjaga keamanan dan integritas data [8]. Ini memainkan peran penting dalam sistem informasi dengan menyediakan fondasi untuk aplikasi yang memanfaatkannya, memungkinkan integrasi data dari berbagai sumber, dan

menyediakan cara yang efektif untuk mengakses dan memanfaatkan informasi yang disimpan.

Adapun tujuan dari basis data yang efektif yaitu [1]:

1. Memastikan bahwa setiap data yang disimpan di dalam basis data dapat digunakan atau berbagi pakai untuk berbagai aplikasi.
2. Memelihara dan memastikan keakuratan serta konsistensi data yang disimpan di dalam basis data.
3. Memungkinkan basis data untuk dapat berkembang sesuai dengan kebutuhan pemakai.
4. Memastikan bahwa semua data yang diperlukan dapat tersedia dengan baik pada masa sekarang maupun pada masa yang akan datang.

Terdapat berbagai jenis basis data yang digunakan dalam berbagai konteks, tergantung pada kebutuhan dan karakteristik aplikasi yang akan digunakan. Berikut adalah beberapa jenis basis data yang paling umum [8]:

1. Basis Data Relasional

   Basis data ini merupakan basis data yang paling umum dan sering digunakan. Data disimpan dalam tabel yang terdiri dari baris dan kolom, di mana relasi antara data direpresentasikan oleh kunci primer dan kunci asing. Contoh dari sistem basis data relasional adalah MySQL, PostgreSQL, Oracle, dan SQL Server.

2. Basis Data Non-Relasional

   Basis data Non-Relasional ini memungkinkan penyimpanan objek dan atributnya dalam satu entitas tunggal, yang memungkinkan penggunaan konsep pemrograman berorientasi objek secara langsung dalam penyimpanan dan pengambilan data. Contoh basis data berorientasi objek adalah MongoDB dan Couchbase.